ETFs Project Requirements

Evaluation Criteria

- Does the application provide all the required functionality? Does the page include the sections requested? Is the date current? Is there a way to select different search criteria? Does the data change when different criteria are selected? Are the results presented in a logical and useful form?

- Does the application use Angular correctly to produce a Single Page Application? Are asynchronous requests used? Is functionality appropriately split between different classes? Is the interaction between classes implemented appropriately? Does the code demonstrate TypeScript coding best practices (e.g., type-safety, DRY)?

- Does the application use HTML5 semantic tags properly? Is the presentation split between the different sections as requested? Does the application use CSS functionality to construct a consistent style applied to all the sections on the web page?

- Does the solution follow testing best practices? Are there sufficient unit tests to provide effective coverage of the functionality? Are unit tests isolated from other program units? Do they all pass? Are there negative tests? Are there E2E tests? Do they all pass?

Tasks

Create a dynamic web application that displays a list of ETFs (Exchanged-Traded Funds). The user must be able to choose a criterion such as total return or expense ratio that will determine what ETFs will be in the list that is displayed.

Create E2E test for at least the following two scenarios:

1. Navigate to the main page and verify the initial list of ETFs.
2. Click a button and verify that the list of ETFs is displayed in a different order.

Requirements

- You will be provided with a starter project that provides the web service (described below). Your instructor will tell you how to run this service. The index page of the service shows how to request different criteria and which sorting criteria are available.

- Your client-side code will make requests to the web service and display the list that is returned. The application must provide buttons, links, etc., for all the sorting criteria that are documented on the service index page.

- Numeric values and currency values must be formatted consistently; for example, right justified, with the same number of digits after the decimal point.

- The application must be developed in a Single Page Application style using Angular. In other words, asynchronous requests should be sent to the service and the results used to update the web page.

- Your web page layout must be defined with HTML5 semantic tags. The page should contain a header that displays the name of the application, a footer that contains the site name and current date, a nav section that provides the user the option to make the various requests for data that are sent to the web service, and a content section that displays the data. Display the information in the form that seems most natural and useful.

- The website that you build should illustrate HMTL, CSS, and Angular best practices. The content (HTML) and style (CSS) should be clearly separated. The navigation should be intuitive and demonstrate best practices in that area.

- Angular functionality should be appropriately split between components and services. Each unit of Angular code that you create (components, services, etc.) should have suitable unit tests that pass. Remember to write both positive and negative tests.

ETF REST Service

The ETF REST service is implemented as a Node.js web application that is deployed on your RVC.

To install the ETF service:

1. Extract `M:\FSE\WebApps\ETFs.zip` to your `D:\`
2. Open a command prompt in `D:\ETFs` and run the following commands:
    a. `npm install`
    b. `npm start`
3. Browse to the service's landing page at http://localhost:8080/
4. Note the links with URLs to fetch ETFs sorted in various orders.